



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/617,592	07/11/2003	David C. Kung	124263-1013	3712
<div>7590 01/18/2007 Gardere Wynne Sewell LLP 3000 Thanksgiving Tower Suite 3000 1601 Elm Street Dallas, TX 75201-4767</div>			<div>EXAMINER YIGDALL, MICHAEL J</div>	
			<div>ART UNIT 2192</div>	<div>PAPER NUMBER</div>
SHORTENED STATUTORY PERIOD OF RESPONSE		MAIL DATE	DELIVERY MODE	
3 MONTHS		01/18/2007	PAPER	

**Please find below and/or attached an Office communication concerning this application or proceeding.**

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

# Office Action Summary

Application No.

10/617,592

Applicant(s)

KUNG ET AL.

Examiner

Michael J. Yigdall

Art Unit

2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

- 1) ☒ Responsive to communication(s) filed on 11 July 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

- 4) ☒ Claim(s) 1-28 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-28 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

## Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 11 July 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

## Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO/SB/08)  
Paper No(s)/Mail Date 12/24/03.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: \_\_\_\_\_

### **DETAILED ACTION**

1. Claims 1-28 are pending. A priority date of July 11, 2003 is considered.

#### ***Specification***

2. The use of the trademark "JAVA" has been noted in this application (see, for example, claim 12). It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner that might adversely affect their validity as trademarks.

#### ***Claim Objections***

3. Claim 8 is objected to because of the following informalities: The claim includes the abbreviation "API" with no corresponding definition recited in the claims. Appropriate correction is required.

#### ***Claim Rejections - 35 USC § 112***

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claim 10 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter that Applicant regards as the invention.

With respect to claim 10, the claim recites, “the sequence diagram displays the condition of a method call,” which lacks antecedent basis in the claim. As recited, claim 10 is dependent on claim 1, and claim 1 does not introduce a sequence diagram. Thus, there is insufficient antecedent basis for “the sequence diagram” in claim 10. The examiner presumes that claim 10 is to be considered dependent on claim 9, which does recite a sequence diagram.

***Claim Rejections - 35 USC § 101***

6. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

7. Claims 16-28 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

With respect to claims 16-21, the claims are directed to a system that amounts to software *per se*. There are no computer or hardware components recited in the claims that would permit the functionality of the system to be realized. Accordingly, the claims are directed to non-statutory subject matter. See MPEP § 2106.01.

With respect to claims 22-28, the claims are directed to an apparatus that amounts to software *per se*. There are no computer or hardware components recited in the claims that would permit the functionality of the apparatus to be realized. Accordingly, the claims are directed to non-statutory subject matter. See MPEP § 2106.01.

***Claim Rejections - 35 USC § 103***

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

9. Claims 1, 9, 11-17, 22 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,851,107 to Coad et al. ("Coad") in view of U.S. Patent No. 6,598,181 to Pennell ("Pennell").

With respect to claim 1, Coad discloses a process for providing a representation of specified characteristics of a previously developed object-oriented software program (see, for example, column 4, lines 38-41, which shows providing a representation of a program, and column 15, line 61 to column 16, line 4, which further shows that the program was previously developed in an object-oriented language), said program including a number of object classes and further including object related methods belonging to respective classes (see, for example, column 5, lines 14-20, which shows that the program includes object classes and methods).

Coad further discloses sensing the methods included in the software program and measuring the complexity of the object classes (see, for example, column 7, Table 3), but does not expressly disclose said process comprising the steps of:

sensing at least one complex method call included in said software program, a plurality of said methods being associated with each of said complex method calls; and

extracting a number of single method calls from each of said complex method calls.

However, in an analogous art, Pennell discloses sensing at least one complex method call included in a software program, a plurality of methods being associated with the complex method call (see, for example, column 2, line 65 to column 3, line 5). Pennell further discloses extracting a number of single method calls from the complex method call (see, for example, column 5, lines 29-40).

Coad is directed to helping a programmer understand and visualize a program (see, for example, column 1, lines 38-46). Likewise, Pennell discloses that extracting single method calls from complex method calls helps a programmer understand and debug a program (see, for example, column 1, line 21-33 and column 2, lines 41-50).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include in Coad, sensing at least one complex method call included in said software program, a plurality of said methods being associated with each of said complex method calls, and extracting a number of single method calls from each of said complex method calls, so as to further improve program understanding.

Coad in view of Pennell further discloses said process comprising the steps of:

generating a set of information for each of said methods from said single method calls, the information set for a particular method containing at least the name of the particular method and the class to which the particular method belongs (see, for example, Coad, column 4, lines 47-51, which shows generating a model of the program, and column 5, lines 4-20, which further shows that the model includes information for each method and the class to which it belongs, such as in the example illustrated in FIGS. 4 and 5); and

constructing a representation of interactions between objects of said software program from the information contained in said method information sets (see, for example, Coad, column 4, lines 52-54, which shows constructing a representation of the program from the model, and column 17, lines 2-8, which further shows that the representation depicts interactions among objects of the program).

With respect to claim 9, the rejection of claim 1 is incorporated, and Coad in view of Pennell further discloses that:

said constructing step comprises constructing a sequence diagram depicting the interactions between respective objects of said software program (see, for example, Coad, FIG. 14 and column 17, lines 2-8, which shows constructing a sequence diagram that depicts the interactions among objects of the program).

With respect to claim 11, the rejection of claim 1 is incorporated, and Coad in view of Pennell further discloses that:

said software program is in the form of source code (see, for example, Coad, column 15, lines 61-64, which shows that the program is in the form of source code).

With respect to claim 12, the rejection of claim 1 is incorporated, and Coad in view of Pennell further discloses that:

said software program is written in Java software code (see, for example, Coad, column 16, lines 1-4, which shows that the program is written in the Java language).

Art Unit: 2192

With respect to claim 13, the rejection of claim 1 is incorporated, and Coad in view of Pennell further discloses that:

said software program is written in C++ software code (see, for example, Coad, column 16, lines 1-4, which shows that the program is written in the C++ language).

With respect to claim 14, the rejection of claim 1 is incorporated, and Coad in view of Pennell further discloses that:

at least one of said object related methods in said program is a polymorphic method (see, for example, Coad, column 9, Table 8, which shows the polymorphism of methods in the program).

With respect to claim 15, the rejection of claim 1 is incorporated, and Coad in view of Pennell further discloses that:

at least one of said object related methods in said program is related to an inheritance feature, and said extraction step includes tracking an inheritance path until it reaches a parent class wherein the method is defined (see, for example, Coad, column 8, Table 6, which shows tracking an inheritance tree to identify where a class and its methods are defined).

With respect to claim 16, Coad discloses a system for providing a representation of specified characteristics of a previously developed object-oriented software program (see, for example, column 4, lines 38-41, which shows providing a representation of a program, and column 15, line 61 to column 16, line 4, which further shows that the program was previously developed in an object-oriented language), said program including a number of object classes

Art Unit: 2192

and object related single methods belonging to respective classes (see, for example, column 5, lines 14-20, which shows that the program includes object classes and methods).

Coad further discloses that the program includes method calls that affect the complexity of the classes (see, for example, column 7, Table 3), but does not expressly disclose said program further including at least one complex method call containing a plurality of said single methods, said system comprising:

a Method Detail Parser unit disposed to extract a number of individual method calls from each of said complex method calls.

However, in an analogous art, Pennell discloses a software program that includes at least one complex method call containing a plurality of single methods (see, for example, column 2, line 65 to column 3, line 5). Pennell further discloses extracting a number of individual method calls from the complex method call (see, for example, column 5, lines 29-40).

Coad is directed to helping a programmer understand and visualize a program (see, for example, column 1, lines 38-46). Likewise, Pennell discloses that extracting individual method calls from complex method calls helps a programmer understand and debug a program (see, for example, column 1, line 21-33 and column 2, lines 41-50).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include in Coad, a Method Detail Parser unit disposed to extract a number of individual method calls from each of at least one complex method calls containing a plurality of said single methods, so as to further improve program understanding.

Coad in view of Pennell further discloses said system comprising:

a data base operable to store a set of information for each of said single methods, the information set for a particular single method containing at least the name of the particular method and the class to which the particular method belongs (see, for example, Coad, column 4, lines 47-51, which shows storing a model of the program, and column 5, lines 4-20, which further shows that the model includes information for each method and the class to which it belongs, such as in the example illustrated in FIGS. 4 and 5); and

a drawing device operable to construct a representation of interactions between objects of said software program from the information contained in said method information sets (see, for example, Coad, column 4, lines 52-54, which shows constructing a representation of the program from the model, and column 17, lines 2-8, which further shows that the representation depicts interactions among objects of the program).

With respect to claim 17, the rejection of claim 16 is incorporated, and Coad in view of Pennell further discloses that:

said system includes a Method Information Parser unit disposed to extract the name and class of each of said single methods from said software program (see, for example, Coad, column 5, lines 4-20, which shows extracting the name and class of each method to include in the model, such as in the example illustrated in FIGS. 4 and 5).

With respect to claim 22, Coad discloses an apparatus for providing a sequence diagram representing specified characteristics of a previously developed object-oriented software program (see, for example, column 4, lines 38-41, which shows providing a representation of a program, and column 15, line 61 to column 16, line 4, which further shows that the program was

Art Unit: 2192

previously developed in an object-oriented language, and see, for example, FIG. 14 and column 17, lines 2-8, which shows that the representation is a sequence diagram), said program including a number of object classes and further including object related methods belonging to respective classes (see, for example, column 5, lines 14-20, which shows that the program includes object classes and methods).

Coad further discloses sensing the methods included in the software program and measuring the complexity of the object classes (see, for example, column 7, Table 3), but does not expressly disclose said apparatus comprising:

means for sensing at least one complex method call included in said software program, a plurality of said methods being associated with each of said complex method calls;

Method Detail Parser means for extracting a number of single method calls from each of said complex method calls.

However, in an analogous art, Pennell discloses sensing at least one complex method call included in a software program, a plurality of methods being associated with the complex method call (see, for example, column 2, line 65 to column 3, line 5). Pennell further discloses extracting a number of single method calls from the complex method call (see, for example, column 5, lines 29-40).

Coad is directed to helping a programmer understand and visualize a program (see, for example, column 1, lines 38-46). Likewise, Pennell discloses that extracting single method calls from complex method calls helps a programmer understand and debug a program (see, for example, column 1, line 21-33 and column 2, lines 41-50).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include in Coad, means for sensing at least one complex method call included in said software program, a plurality of said methods being associated with each of said complex method calls, and Method Detail Parser means for extracting a number of single method calls from each of said complex method calls, so as to further improve program understanding.

Coad in view of Pennell further discloses said apparatus comprising:

means for generating a set of information for each of said object related methods from said single method calls, the information set for a particular object related method containing at least the name of the particular method and the class to which the particular method belongs (see, for example, Coad, column 4, lines 47-51, which shows generating a model of the program, and column 5, lines 4-20, which further shows that the model includes information for each method and the class to which it belongs, such as in the example illustrated in FIGS. 4 and 5); and

means for constructing sequence diagram representing interactions between objects of said software program from the information contained in said method information sets (see, for example, Coad, column 4, lines 52-54, which shows constructing a representation of the program from the model, and FIG. 14 and column 17, lines 2-8, which further shows that the representation is a sequence diagram that depicts interactions among objects of the program).

With respect to claim 23, the rejection of claim 22 is incorporated, and Coad in view of Pennell further discloses that:

said apparatus includes Method Information Parser means for extracting the name and class of each of said methods from said software program (see, for example, Coad, column 5,

Art Unit: 2192

lines 4-20, which shows extracting the name and class of each method to include in the model, such as in the example illustrated in FIGS. 4 and 5).

10. Claims 2-8, 18-21 and 24-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Coad in view of Pennell, as applied to claims 1, 17 and 23 above, respectively, and further in view of U.S. Pub. No. 2003/0188299 to Broughton et al. ("Broughton").

With respect to claim 2, the rejection of claim 1 is incorporated. Coad in view of Pennell does not expressly disclose that:

said extracting step comprises replacing a component of a complex method call with a phase variable to produce a method call of reduced complexity.

Nonetheless, Coad suggests replacing a component of a complex assignment statement with a phase variable to produce an assignment statement of reduced complexity (see, for example, FIGS. 8B and 8C).

Furthermore, in an analogous art, Broughton discloses replacing a component of a complex method call with a phase variable to produce a method call of reduced complexity (see, for example, paragraphs [0068]-[0070], which shows replacing a component "a & b & c" of a complex method call "decrement (d, a & b & c)" with a phase variable "temp = a & b & c" to produce a method call of reduced complexity "decrement (d, temp)"). Broughton discloses that such an expansion or extraction step facilitates later binding and aliasing optimizations (see, for example, paragraph [0071]).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Coad and Pennell such that said extracting step comprises

Art Unit: 2192

replacing a component of a complex method call with a phase variable to produce a method call of reduced complexity, as Coad suggests, so as to facilitate later binding and aliasing optimizations, as Broughton teaches.

With respect to claim 3, the rejection of claim 2 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said process includes an initial step of extracting the name and class of each of said methods from said software program (see, for example, Coad, column 5, lines 4-20, which shows extracting the name and class of each method to include in the model, such as in the example illustrated in FIGS. 4 and 5).

With respect to claim 4, the rejection of claim 1 is incorporated, and Coad in view of Pennell further discloses that:

a given complex method call comprises multiple method related components (see, for example, Pennell, column 2, line 65 to column 3, line 5, which shows that the complex method call “total (get first value, get second value)” comprises three method related components).

Coad in view of Pennell does not expressly disclose that:

said extracting step comprises recursively substituting a phase variable for each of said method related components, until said given complex method call has been resolved into multiple lines, each containing one of said single method calls.

Nonetheless, Coad suggests recursively substituting a phase variable for each assignment related component of a complex assignment statement, until the complex assignment statement

Art Unit: 2192

has been resolved into multiple lines, each containing one of the assignment related components (see, for example, FIGS. 8B and 8C).

Furthermore, in an analogous art, Broughton discloses recursively substituting a phase variable for each component of a complex method call, until the complex method call has been resolved into multiple lines (see, for example, paragraphs [0068]-[0070], which shows substituting a phase variable “temp = a & b & c” for a component “a & b & c” of a complex method call “decrement (d, a & b & c)” to resolve the complex method call into multiple lines). Broughton discloses that such an expansion or extraction step facilitates later binding and aliasing optimizations (see, for example, paragraph [0071]).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Coad and Pennell such that said extracting step comprises recursively substituting a phase variable for each of said method related components, until said given complex method call has been resolved into multiple lines, each containing one of said single method calls, as Coad suggests, so as to facilitate later binding and aliasing optimizations, as Broughton teaches.

With respect to claim 5, the rejection of claim 4 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that said extracting step comprises:

a first parsing phase disposed to separate any casting operations included in said given complex method call (see, for example, Coad, column 14, lines 43-44, which shows separating any casting operations, such as casting operations that are considered unnecessary);

a second parsing phase disposed to isolate any method parameters included in said given complex method call (see, for example, Pennell, column 2, line 65 to column 3, line 5, which shows isolating any method parameters, such as the method parameter “get first value”); and

a third parsing phase disposed to resolve any continuous method calls included in said given complex method call into multiple lines, each containing one of said single method calls (see, for example, Pennell, column 5, lines 1-40, which shows resolving any continuous method calls into multiple lines, such as the multiple “call” instructions).

With respect to claim 6, the rejection of claim 5 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said first parsing phase is implemented prior to said second parsing phase, and said second parsing phase is implemented prior to said third parsing phase (see, for example, Coad, FIG. 9, which shows that the parsing phases are implemented in sequence at step 906).

With respect to claim 7, the rejection of claim 6 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said step of generating method information sets includes parsing an output provided by said third parsing phase to determine the correct object class for each of said object related methods (see, for example, Coad, column 16, lines 13-16, which shows parsing the program to generate the model, and column 5, lines 4-20, which shows that the model identifies the class of each method, such as in the example illustrated in FIGS. 4 and 5).

With respect to claim 8, the rejection of claim 7 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said process includes the step of determining whether a method is a user-defined or a standard API method (see, for example, Coad, column 14, lines 25-28 and 37-43, which shows determining whether a method is a method defined in the source code or an imported method, and column 14, lines 29-33, which further shows identifying standard API methods imported from the “java.lang” package).

With respect to claim 18, the rejection of claim 17 is incorporated. Coad in view of Pennell further discloses that a plurality of method related components are contained in a given complex method call (see, for example, Pennell, column 2, line 65 to column 3, line 5, which shows that the complex method call “total (get first value, get second value)” contains three method related components), but does not expressly disclose that:

said Method Detail Parser unit is disposed to recursively substitute a phase variable for each of a plurality of method related components contained in a given complex method call, until said given complex method call has been resolved into multiple lines, each containing one of said single method calls.

Nonetheless, Coad suggests recursively substituting a phase variable for each assignment related component of a complex assignment statement, until the complex assignment statement has been resolved into multiple lines, each containing one of the assignment related components (see, for example, FIGS. 8B and 8C).

Furthermore, in an analogous art, Broughton discloses recursively substituting a phase variable for each component of a complex method call, until the complex method call has been resolved into multiple lines (see, for example, paragraphs [0068]-[0070], which shows substituting a phase variable “temp = a & b & c” for a component “a & b & c” of a complex

Art Unit: 2192

method call “decrement (d, a & b & c)” to resolve the complex method call into multiple lines).

Broughton discloses that such an expansion or extraction step facilitates later binding and aliasing optimizations (see, for example, paragraph [0071]).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Coad and Pennell such that said Method Detail Parser unit is disposed to recursively substitute a phase variable for each of a plurality of method related components contained in a given complex method call, until said given complex method call has been resolved into multiple lines, each containing one of said single method calls, as Coad suggests, so as to facilitate later binding and aliasing optimizations, as Broughton teaches.

With respect to claim 19, the rejection of claim 18 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said Method Detail Parser unit is sequentially operated to implement a first parsing phase to separate any casting operations included in said given complex method call (see, for example, Coad, column 14, lines 43-44, which shows separating any casting operations, such as casting operations that are considered unnecessary), to implement a second parsing phase to isolate any method parameters included in said given complex method call (see, for example, Pennell, column 2, line 65 to column 3, line 5, which shows isolating any method parameters, such as the method parameter “get first value”), and to implement a third parsing phase to resolve said given complex method call into multiple lines, each containing one of said single method calls (see, for example, Pennell, column 5, lines 1-40, which shows resolving any continuous method calls into multiple lines, such as the multiple “call” instructions).

With respect to claim 20, the rejection of claim 19 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said drawing device is operable to construct a sequence diagram depicting the interactions between respective objects of said software program (see, for example, Coad, FIG. 14 and column 17, lines 2-8, which shows constructing a sequence diagram that depicts the interactions among objects of the program).

With respect to claim 21, the rejection of claim 20 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said software program is in the form of source code (see, for example, Coad, column 15, lines 61-64, which shows that the program is in the form of source code).

With respect to claim 24, the rejection of claim 23 is incorporated. Coad in view of Pennell further discloses that a plurality of method related components are contained in a given complex method call (see, for example, Pennell, column 2, line 65 to column 3, line 5, which shows that the complex method call “total (get first value, get second value)” contains three method related components), but does not expressly disclose that:

said Method Detail Parser means is operable to recursively substitute a phase variable for each of a plurality of method related components contained in a given complex method call, until said given complex method call has been resolved into multiple lines, each containing one of said single method calls.

Nonetheless, Coad suggests recursively substituting a phase variable for each assignment related component of a complex assignment statement, until the complex assignment statement

Art Unit: 2192

has been resolved into multiple lines, each containing one of the assignment related components (see, for example, FIGS. 8B and 8C).

Furthermore, in an analogous art, Broughton discloses recursively substituting a phase variable for each component of a complex method call, until the complex method call has been resolved into multiple lines (see, for example, paragraphs [0068]-[0070], which shows substituting a phase variable “temp = a & b & c” for a component “a & b & c” of a complex method call “decrement (d, a & b & c)” to resolve the complex method call into multiple lines). Broughton discloses that such an expansion or extraction step facilitates later binding and aliasing optimizations (see, for example, paragraph [0071]).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Coad and Pennell such that said Method Detail Parser means is operable to recursively substitute a phase variable for each of a plurality of method related components contained in a given complex method call, until said given complex method call has been resolved into multiple lines, each containing one of said single method calls, as Coad suggests, so as to facilitate later binding and aliasing optimizations, as Broughton teaches.

With respect to claim 25, the rejection of claim 24 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said Method Detail Parser means is disposed to separate any casting operations included in said given complex method call during a first parsing phase (see, for example, Coad, column 14, lines 43-44, which shows separating any casting operations, such as casting operations that are considered unnecessary), to isolate any method parameters included therein during a second parsing phase (see, for example, Pennell, column 2, line 65 to column 3, line 5, which shows

Art Unit: 2192

isolating any method parameters, such as the method parameter “get first value”), and resolve said given complex method call into multiple lines, each containing one of said single method calls, during a third parsing phase (see, for example, Pennell, column 5, lines 1-40, which shows resolving any continuous method calls into multiple lines, such as the multiple “call” instructions).

With respect to claim 26, the rejection of claim 25 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said first parsing phase is implemented prior to said second parsing phase, and said second parsing phase is implemented prior to said third parsing phase (see, for example, Coad, FIG. 9, which shows that the parsing phases are implemented in sequence at step 906).

With respect to claim 27, the rejection of claim 26 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said constructing means comprises a drawing engine for depicting interactions between respective objects of said software program (see, for example, Coad, column 17, lines 2-8, which shows depicting interactions among objects of the program).

With respect to claim 28, the rejection of claim 27 is incorporated, and Coad in view of Pennell in view of Broughton further discloses that:

said software program is in the form of source code (see, for example, Coad, column 15, lines 61-64, which shows that the program is in the form of source code).

Art Unit: 2192

11. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Coad in view of Pennell, as applied to claim 9 above, and further in view of the *OMG Unified Modeling Language Specification Version 1.3* ("OMG").

With respect to claim 10, the rejection of claim 9 is incorporated. Coad in view of Pennell does not expressly disclose that:

the sequence diagram displays the condition of a method call to indicate that the call occurs only when the condition is evaluated to be true.

Nonetheless, Coad suggests that the sequence diagram conforms to the Unified Modeling Language (see, for example, column 15, lines 50-57). It is well known in the art that sequence diagrams conforming to the Unified Modeling Language (UML) display the condition of a method call to indicate that the call occurs only when the condition is evaluated to be true. For example, OMG illustrates a UML sequence diagram that displays the condition "[x > 0]" of a method call "foo(x)" to indicate that the call occurs only when the condition is evaluated to be true (see, for example, Figure 3-48 on page 3-97).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made that the sequence diagram of Coad displays the condition of a method call to indicate that the call occurs only when the condition is evaluated to be true.

### ***Conclusion***

12. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure (see the attached Notice of References Cited).

Art Unit: 2192

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707.

The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

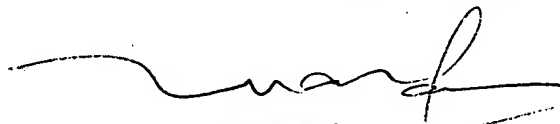
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall  
Examiner  
Art Unit 2192

mjy

  
TUAN DAM  
SUPERVISORY PATENT EXAMINER